XISM-13 (Ref) "Immerse Storage Management
[connection] with the Xanadu Storage Engine"
Ref 3287)

SUBMITTED to BYTE
with Technical Chapters four,
8 Aug 87.

REF
[wh corrections]

XISM McD 2
7 Aug 87

1

IMMENSE STORAGE MANAGEMENT

with

the Xanadu* Storage Engine

Submitted for the BYTE special issue on

Managing Megastorage.

Theodor H. Nelson, Director
Project Xanadu
8480 Fredericksburg #138
San Antonio TX 78229

(c) 1987 T. Nelson

Offering first rights only.

* "Xanadu" is a trademark for software and hypermedia designs and products offered by Project Xanadu, and licensed also to XOC, Incorporated.

XISM MeD 2
7 Aug 87

IMMENSE STORAGE MANAGEMENT

with

the Xanadu* Storage Engine

Theodor H. Nelson, Director
Project Xanadu
8480 Fredericksburg #138
San Antonio TX 78229

(c) 1987 T. Nelson

Offering first rights only.

* "Xanadu" is a trademark for software and hypermedia designs and products offered by Project Xanadu, and licensed also to XOC, Incorporated.

Note to BYTE Magazine:

I am working in MacDraw, and must substitute caps for italics):

This manuscript section is being submitted as the opening and closing notes to bracket Chapters Four (the technical chapters) of my book LITERARY MACHINES, which you are free to revise as you see fit.

The submission has four parts:

pages 1 through 10, with a colored stripe at the top for your convenient recognition. These are text and pictures intended to PRECEDE the material from Literary Machines;

the TEXT of Chapters Four of Literary Machines (done on the PC);

the PICTURES of Chapters Four of Literary Machines (done on the Mac);

and pages N-3 through N, which are meant to FOLLOW the material from Literary Machines (color-striped again for your convenience).

Note to BYTE Magazine:

I am working in MacDraw, and must substitute caps for italics).

This manuscript section is being submitted as the opening and closing notes to bracket Chapters Four (the technical chapters) of my book LITERARY MACHINES, which you are free to revise as you see fit.

The submission has four parts:

pages 1 through 10, with a colored stripe at the top for your convenient recognition. These are text and pictures intended to PRECEDE the material from Literary Machines;

the TEXT of Chapters Four of Literary Machines (done on the PC);

the PICTURES of Chapters Four of Literary Machines (done on the Mac);

and pages N-3 through N, which are meant to FOLLOW the material from Literary Machines (color-striped again for your convenience).

Project Xanadu has suffered from too much publicity. The project is well known, but not well understood. Its greatest aspiration, a universal instantaneous hypertext publishing network*, has not been generally understood at the technical level, and has created various false impressions. (The WHOLE EARTH REVIEW referred to it as "a database-to-be the size of the world"-- May 1985, p. 50-- a very muddled description.)

---

* For more on the hypertext concept, see my recent/forthcoming article in BYTE...

The same software and underlying ideas, however, have another great aspiration: the organization and clarification of files on a small scale, cleaning up the clutter of computer files that now chokes both individuals and offices. People everywhere are drowning in little files with mysterious names whose origins and relationships are forgotten. We want to clear that up as well.

We believe we have a unifying technical solution to both problems.

A NEW FORM OF STORAGE

We have a new model of storage which is both rational and radical: rational because it proposes to keep things orderly to a degree they could never be before; and radical because it requires a fundamental change in the way computers are programmed. Like other new paradigms, this presents an entirely new world-view, and provokes various forms of confusion and anger.

Project Xanadu has suffered from too much publicity. The project is well known, but not well understood. Its greatest aspiration, a universal instantaneous hypertext publishing network*, has not been generally understood at the technical level, and has created various false impressions. (The WHOLE EARTH REVIEW referred to it as "a database-to-be the size of the world"-- May 1985, p. 50-- a very muddled description.)

---

* For more on the hypertext concept, see my recent/forthcoming article in BYTE...

The same software and underlying ideas, however, have another great aspiration: the organization and clarification of files on a small scale, cleaning up the clutter of computer files that now chokes both individuals and offices. People everywhere are drowning in little files with mysterious names whose origins and relationships are forgotten. We want to clear that up as well.

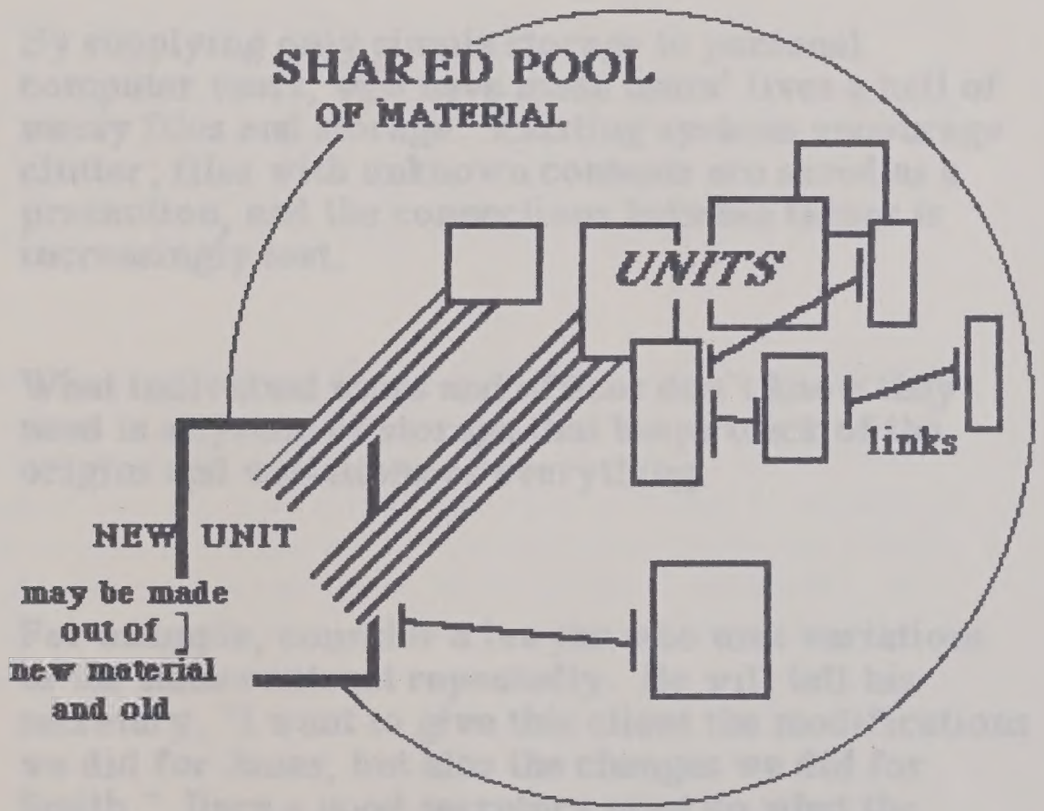
We believe we have a unifying technical solution to both problems.

A NEW FORM OF STORAGE

We have a new model of storage which is both rational and radical: rational because it proposes to keep things orderly to a degree they could never be before; and radical because it requires a fundamental change in the way computers are programmed. Like other new paradigms, this presents an entirely new world-view, and provokes various forms of confusion and anger.

Our generic name for this is "xanalogical storage" (not a trademark). Xanalogical storage alows units to be built from parts of other units, and linked together in various ways. We will explain this concept further, and then present various technical details-- the addressing system and its arithmetic, and the structure of links-- by which we create the particular xanalogical structure of the Xanadu system.

XANALOGICAL STORAGE



Our generic name for this is "xanalogical storage" (not a trademark). Xanalogical storage allows units to be built from parts of other units, and linked together in various ways. We will explain this concept further, and then present various technical details--- the addressing system and its arithmetic, and the structure of links--- by which we create the particular xanalogical structure of the Xanadu system.

XANALOGICAL STORAGE

Apparently simple designs often foist complexity on users, requiring many adaptations OUTSIDE the design. A simple system may push its real problems outside its boundaries, to be dealt with by other means.

Traditional computer storage is such a system; it has been oversimplified by tradition and created many other traditions and complications. Conventional file storage pushes onto the user the problem of naming, and keeping track of, hundreds or thousands of files and their backup copies, and the relations between them.

By supplying only simple storage to personal computer users, we have made users' lives a hell of messy files and storage. Existing systems encourage clutter; files with unknown contents are saved as a precaution, and the connections between things is increasingly lost.

What individual users and offices don't know they need is a system of storage that keeps track of the origins and variations of everything.

For example, consider a lawyer who uses variations of the same contract repeatedly. He will tell his secretary, "I want to give this client the modifications we did for Jones, but also the changes we did for Smith." Here a good secretary must do what the system ought to do-- keep track of the origins and interconnections of the material.

Apparently simple designs often foist complexity on users, requiring many adaptations OUTSIDE the design. A simple system may push its real problems outside its boundaries, to be dealt with by other means.

Traditional computer storage is such a system; it has been oversimplified by tradition and created many other traditions and complications. Conventional file storage pushes onto the user the problem of naming, and keeping track of, hundreds or thousands of files and their backup copies, and the relations between them.

By supplying only simple storage to personal computer users, we have made users' lives a hell of messy files and storage. Existing systems encourage clutter; files with unknown contents are saved as a precaution, and the connections between things is increasingly lost.

What individual users and offices don't know they need is a system of storage that keeps track of the origins and variations of everything.

For example, consider a lawyer who uses variations of the same contract repeatedly. He will tell his secretary, "I want to give this client the modifications we did for Jones, but also the changes we did for Smith." Here a good secretary must do what the system ought to do-- keep track of the origins and interconnections of the material.

I also argued in a recent article ("The Tyranny of the File," DATAMATION, 15 December 1986) that traditional file structures generated many computer applications as we now know them.

Conventional files are streams of bits divided into blocks and given a name. Text systems began when someone decided to treat the sequence of bits as text characters, then set up controls for their revision. Database programs began when someone decided to treat individual blocks as units, and divide them into named and addressable fields.

In this way, I would argue, our application programs have come to be artificially divided into functions on the basis of how they are implemented. This keeps us misled into believing such programs as "database" and "text" are divinely ordained, and leads us away from designing functions best suited to particular uses.

The alternative is a SERIOUS module for maintaining storage in its real complexity, permitting the user to understand the interrelation of all stored materials. Under our storage paradigm it is possible to ascertain the origin of every part of every document (in a way that will even save space in some environments), as well as to make note of every interconnection and feature of interest.

I also argued in a recent article ("The Tyranny of the File," DATAMATION, 15 December 1976) that traditional file structures generated many computer applications as we now know them.

Conventional files are streams of bits divided into blocks and given a name. Text systems began when someone decided to treat the sequence of bits as text characters, then set up controls for their revision. Database programs began when someone decided to treat individual blocks as units, and divide them into named and addressable fields.

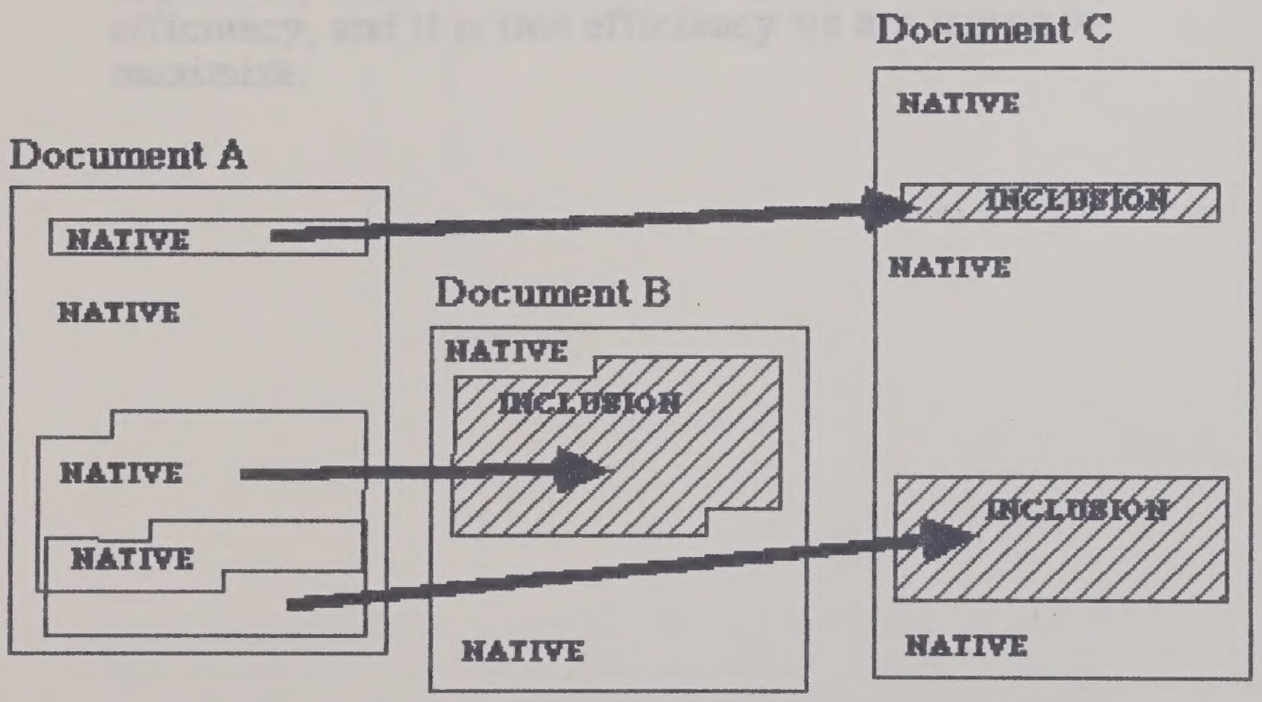
In this way, I would argue, our application programs have come to be artificially divided into functions on the basis of how they are implemented. This keeps us misled into believing such programs as "database" and "text" are divinely ordained, and leads us away from designing functions best suited to particular users.

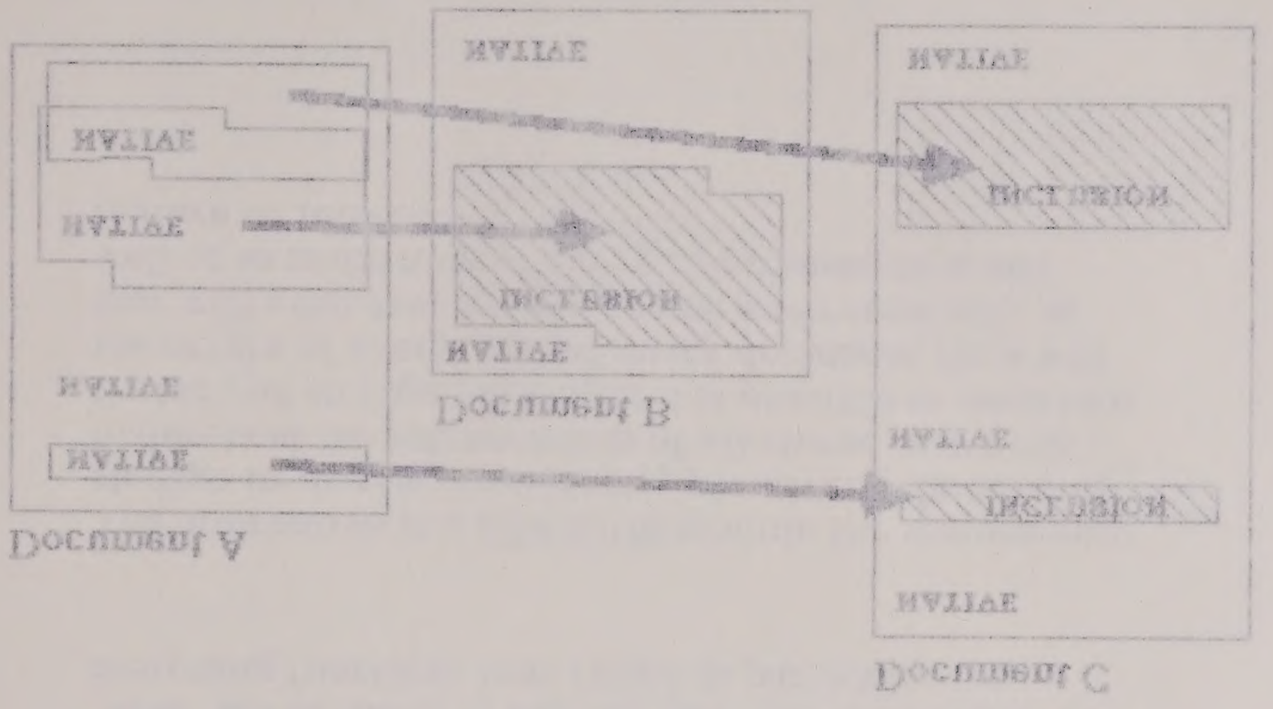
The alternative is a SERIOUS module for maintaining storage in its real complexity, permitting the user to understand the interrelation of all stored materials. Under our storage paradigm it is possible to ascertain the origin of every part of every document (in a way that will even save space in some environments), as well as to make note of every interconnection and feature of interest.

(Note that to distinguish materials stored in this way, we will avoid confusion by not using the word "file," since that is what conventionally stored units are called. We will use another term-- nothing peculiar like "xfile," but simply "document"-- for materials stored according to this new paradigm.)

OUR MODEL, PART 1: INCLUSION

In the Xanadu paradigm, a document consists of NATIVE BYTES (which originated with this document) and INCLUSIONS, which are bytes native to other documents but also present by "inclusion," or virtual copy, in this one. Conceptually, there is only one copy of every byte (though for both safety and implementation there are generally other copies).




NATIVE
NATIVE
NATIVE
NATIVE
INCLUSION
Document A
NATIVE
NATIVE
INCLUSION
Document B
NATIVE
NATIVE
INCLUSION
Document C

A document's content bytes consist of its native bytes and its included bytes. A byte is just as fully a part of a document in which it is included as the document to which it is native.

Thus a document consists (first approximation) of native bytes and a structure of pointers for bringing instances of included bytes from other documents.

By various system commands, the user may ask where bytes came from, ask to see them in their original context, and so on.

At about this point some people invariably ask, "What about efficiency?" Well, at every stage in the development of system facilities, some people object to a lack of raw access. But if we keep in mind that storage and computers are cheap, and people are expensive, then the real efficiency is HUMAN efficiency, and it is that efficiency we are trying to maximize.

A document's content bytes consist of its native bytes and its included bytes. A byte is just as fully a part of a document in which it is included as the document to which it is native.

Thus a document consists (first approximation) of native bytes and a structure of pointers for bringing instances of included bytes from other documents.

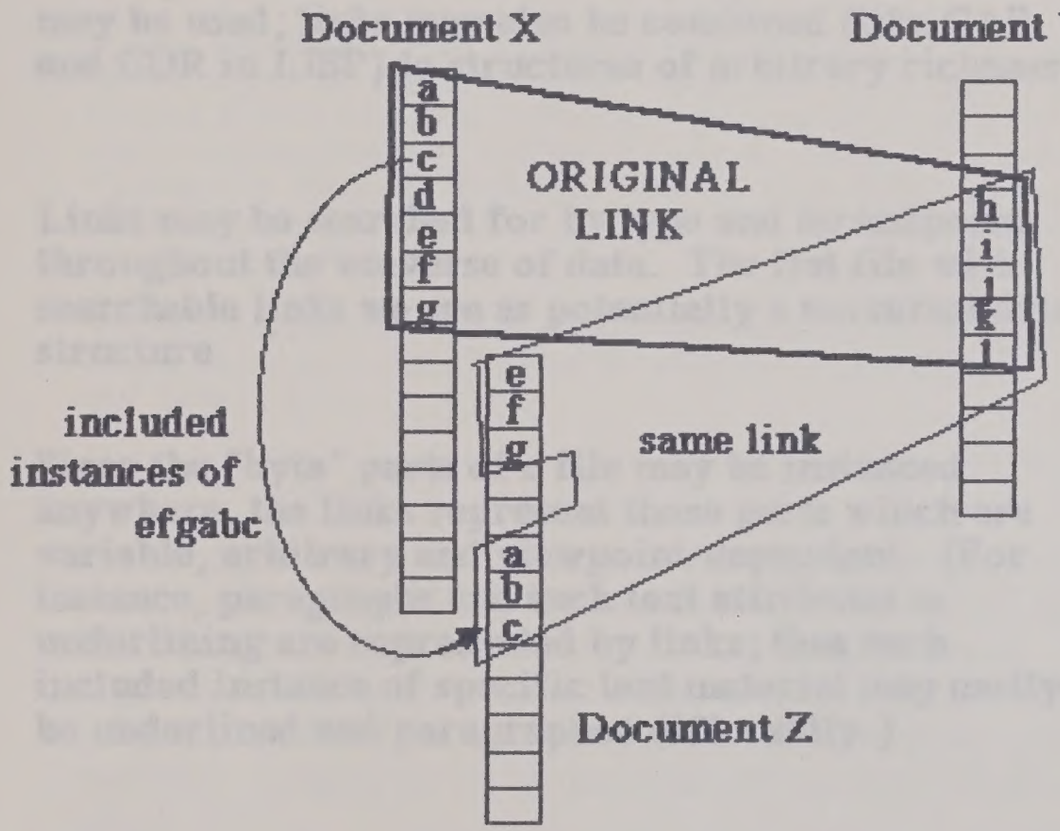
By various system commands, the user may ask where bytes came from, ask to see them in their original context, and so on.

At about this point some people invariably ask, "What about efficiency?" Well, at every stage in the development of system facilities, some people object to a lack of raw access. But if we keep in mind that storage and computers are cheap, and people are expensive, then the real efficiency is HUMAN efficiency, and it is that efficiency we are trying to maximize.

OUR MODEL, PART 2: LINKS

We need a data structure that keeps track of arbitrary links between arbitrary portions of arbitrary documents. This is important for hypertext, for the marking and annotation of all kinds of data, and for searching on such links and markings.

In the Xanadu system we do this by attaching links, not to points in the data, but to the bytes themselves. Thus a given link is present on every document in which any of the bytes linked to is present.



Here, for instance, some of the bytes of Document X have been included into Document Z. A query as to what bytes in Z link to Y will find efg and abc, and the particular link between Z and Y in which they participate.

OUR MODEL, PART 2: LINKS

We need a data structure that keeps track of arbitrary links between arbitrary portions of arbitrary documents. This is important for hypertext, for the marking and annotation of all kinds of data, and for searching on such links and markings.

In the Xanadu system we do this by attaching links, not to points in the data, but to the bytes themselves. Thus a given link is present on every document in which any of the bytes linked to is present.

Here, for instance, some of the bytes of Document X have been included into Document Z. A query as to what bytes in Z link to Y will find efg and abc, and the particular link between Z and Y in which they participate.

GENERALITY

We have taken pains to generalize this so that it can work across all disk, computer and network boundaries as the world of computer storage becomes united. Our storage program is designed to be run in parallel on networks of servers without limit, which respond essentially as a whole.

The link mechanism is totally generalized. Link types are extensible and any user may create new link types. While primitive links are two-sided, one-sided links may be used; links may also be combined (like CAR and CDR in LISP) to structures of arbitrary richness.

Links may be searched for by type and by endpoints throughout the universe of data. The flat file with searchable links we see as potentially a universal data structure.

Since the "byte" parts of a file may be instanced anywhere, the links represent those parts which are variable, arbitrary and viewpoint-dependent. (For instance, paragraphs and such text attributes as underlining are represented by links; thus each included instance of specific text material may easily be underlined and paragraphed differently.)

(GIVEN THIS OVERVIEW, NOW PROCEED TO THE TECHNICAL CHAPTERS OF 'LITERARY MACHINES.)

GENERALITY

We have taken pains to generalize this so that it can work across all disk, computer and network boundaries as the world of computer storage becomes united. Our storage program is designed to be run in parallel on networks of servers without limit, which respond essentially as a whole.

The link mechanism is totally generalized. Link types are extensible and any user may create new link types. While primitive links are two-sided, one-sided links may be used; links may also be combined (like CAR and CDR in LISP) to structures of arbitrary richness.

Links may be searched for by type and by endpoints throughout the universe of data. The flat file with searchable links we see as potentially a universal data structure.

Since the "byte" parts of a file may be instanced anywhere, the links represent those parts which are variable, arbitrary and viewpoint-dependent. (For instance, paragraphs and such text attributes as underlining are represented by links; thus each included instance of specific text material may easily be underlined and paragraphed differently.)

(GIVEN THIS OVERVIEW, NOW PROCEED TO THE TECHNICAL CHAPTERS OF 'LITERARY MACHINES.)

IMPLEMENTATION

The Xanadu storage engine is a program that presently exists as described in full prototype. (It is on the end of a phone line for experimentation by serious developers.) Its software mechanisms are proprietary and presently covered by trade secret;* we hope to publish them at a future date.

---

* The system is based on proprietary mechanisms, not known to the literature in their generality, which we call enfilades (from the French ENFILADE, "in a file" or "succession of units"). Enfilades, their ramifications and different designs, are the discoveries of Theodor H. Nelson, William F. Barus, Mark Miller, Stuart Greene, Roger Gregory and K. Eric Drexler. The design here is constructed from several enfilades.

The present architecture is chiefly the work of Mark S. Miller and Roger Gregory, with Theodor H. Nelson, Stuart Greene, Eric Hill and Roland King.

The program is written in C under Unix. In the current version (August 1987) it compiles to about 135K on the 68000. (A local search space of ten megabytes is desirable, though we expect that to be reducible for personal and office applications.) The resident protocol manager (required by application programs) compiles to about 35K.

This is a large-scale problem where clean design is vital--both inner simplicity and a clean interface to the outside world. We believe we have achieved this.

(CONCLUDING SECTION)

IMPLEMENTATION

The Xanadu storage engine is a program that presently exists as described in full prototype. (It is on the end of a phone line for experimentation by serious developers.) Its software mechanisms are proprietary and presently covered by trade secret,* we hope to publish them at a future date.

---

* The system is based on proprietary mechanisms, not known to the literature in their generality, which we call enfilades (from the French ENFILADE, "in a file" or "succession of units"). Enfilades, their ramifications and different designs, are the discoveries of Theodor H. Nelson, William F. Barns, Mark Miller, Stuart Greene, Roger Gregory and K. Eric Drexler. The design here is constructed from several enfilades.

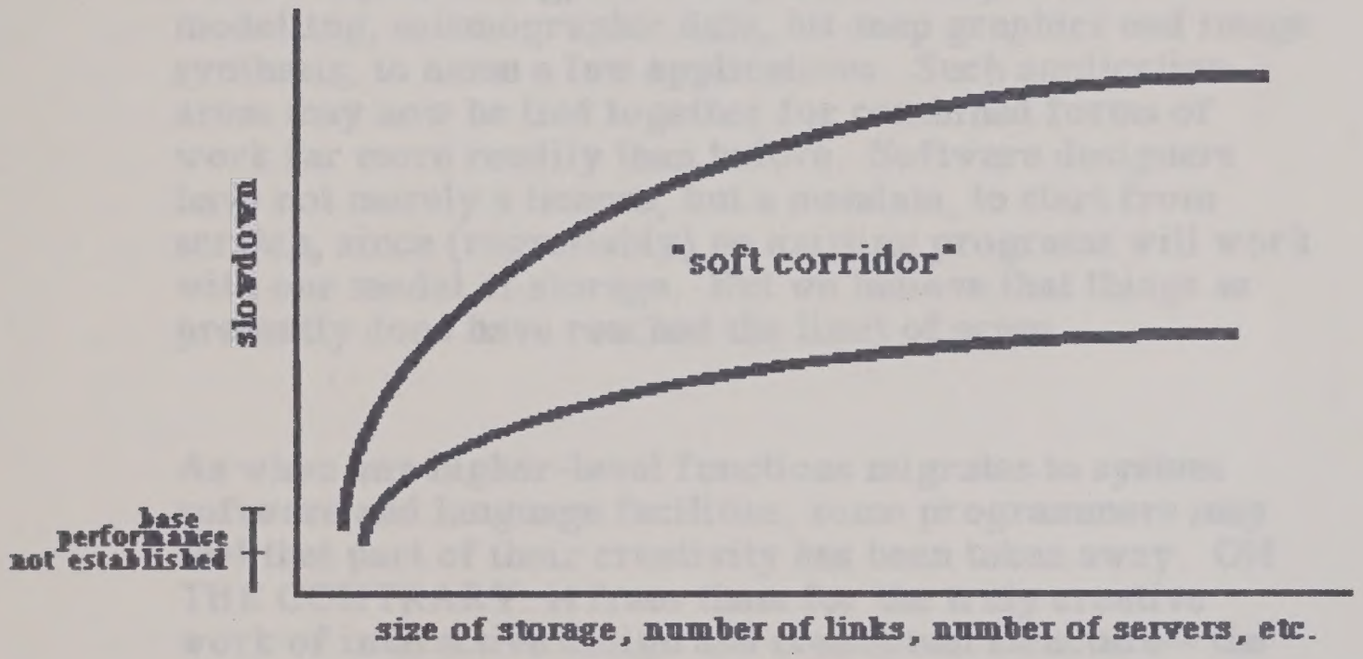
The present architecture is chiefly the work of Mark S. Miller and Roger Gregory, with Theodor H. Nelson, Stuart Greene, Eric Hill and Roland King.

The program is written in C under Unix. In the current version (August 1987) it compiles to about 135K on the 68000. (A local search space of ten megabytes is desirable, though we expect that to be reducible for personal and office applications.) The resident protocol manager (required by application programs) compiles to about 35K.

This is a large-scale problem where clean design is vital—both inner simplicity and a clean interface to the outside world. We believe we have achieved this.

ANTICIPATED PERFORMANCE FUNCTIONS

Our analysis of the algorithms indicates that performance will be loglike (worst case, square-root-like) on all major parameters. For instance, as the number of links doubles, there should be only slight degradations of performace.



Unfortunately this analysis does not give us base rates to judge what performance we will get on presently-available machines, and we await these empirical figures with great interest..

However, the present design is for sequential machines as a convenience. Given our long-term goals and today's new hardware, we believe we can introduce various types of parallelism to improve performance and make the system practical on the scale we intend.

ANTICIPATED PERFORMANCE FUNCTIONS

Our analysis of the algorithms indicates that performance will be loglike (worst case, square-root-like) on all major parameters. For instance, as the number of links doubles, there should be only slight degradations of performance.

Unfortunately this analysis does not give us base rates to judge what performance we will get on presently-available machines, and we await these empirical figures with great interest.

However, the present design is for sequential machines as a convenience. Given our long-term goals and today's new hardware, we believe we can introduce various types of parallelism to improve performance and make the system practical on the scale we intend.

APPLICATION DESIGN

The interior design of applications for use with the Xanadu storage engine is very different from conventional application design, since so much is handled by the storage mechanism. All references to links and stored materials go through the FEBE protocol, and so the programmer's design work becomes to a large extent the user interface.

Note that the application designer is no longer constrained by old categories of program, since the Xanadu data structure provides a broad-spectrum representation method for word processing, database, CAD/CAM, molecular modelling, seismographic data, bit-map graphics and image synthesis, to name a few applications. Such application areas may now be tied together for combined forms of work far more readily than before. Software designers have not merely a license, but a mandate, to start from scratch, since (regrettably) no existing programs will work with our model of storage. But we believe that things as presently done have reached the limit of sense.

As when any higher-level functions migrates to system software and language facilities, some programmers may feel that part of their creativity has been taken away. ON THE CONTRARY: it frees them for the truly creative work of interactive design and conceptual structure-- the virtuality* of mental interaction, screen design. Most application programmers are usually so busy with the impedimenta of storage and data structure maintenance that they tend to sleight the more important subtleties of interaction and virtuality. Now they can concentrate fully on these issues.

* I have argued ("Interactive Systems and the Design of Virtuality," CREATIVE COMPUTING, Nov & Dec 1980, and more recently in COMPUTER LIB, second edition (Microsoft Press, approx. October 1987) that talk of "interfaces" and "metaphors" takes us in the wrong direction, and that thinking about the design of VIRTUALITY— the conceptual structure and feel of a system— a wholly arbitrary new form of mental architecture— leads to better and deeper design.

APPLICATION DESIGN

The interior design of applications for use with the Xanadu storage engine is very different from conventional application design, since so much is handled by the storage mechanism. All references to links and stored materials go through the FEBE protocol, and so the programmer's design work becomes to a large extent the user interface.

Note that the application designer is no longer constrained by old categories of program, since the Xanadu data structure provides a broad-spectrum representation method for word processing, database, CAD/CAM, molecular modelling, seismographic data, bit-map graphics and image synthesis, to name a few applications. Such application areas may now be tied together for combined forms of work far more readily than before. Software designers have not merely a license, but a mandate, to start from scratch, since (regrettably) no existing programs will work with our model of storage. But we believe that things as presently done have reached the limit of sense.

As when any higher-level functions migrates to system software and language facilities, some programmers may feel that part of their creativity has been taken away. ON THE CONTRARY: it frees them for the truly creative work of interactive design and conceptual structure-- the virtuality* of mental interaction, screen design. Most application programmers are usually so busy with the impedimenta of storage and data structure maintenance that they tend to sleight the more important subtleties of interaction and virtuality. Now they can concentrate fully on these issues.

* I have argued ("Interactive Systems and the Design of Virtuality," CREATIVE COMPUTING, Nov & Dec 1984, and more recently in COMPUTER LIB, second edition (Microsoft Press, approx. October 1987) that talk of "interfaces" and "metaphors" takes us in the wrong direction, and that thinking about the design of VIRTUALITY-- the conceptual structure and feel of a system-- a wholly arbitrary new form of mental architecture-- leads to better and deeper design.

THE OVERALL OBJECTIVE: UNIVERSAL HYPERTEXT PUBLISHING

N

These ideas may strain the imagination, but that's good. The unstretched imagination will atrophy.

The grand ambition of the Xanadu system is not "a database the size of the world," but rather a repository publishing network for anybody's documents and contents, which users may combine and link to freely.

This will permit a new form of electronic publishing, entirely within the tradition of paper publishing, but greatly streamlined: one need not ask permission to republish something, but simply place it in a document as an inclusion. Nothing will ever be misquoted or out of context, since the user may inquire as to the origins and native form of any quotation or other inclusion. Royalty will be automatically paid by a user whenever he or she draws out a bytefrom a published document. The permission structure of conventional publishing is bypassed. (Blanket permission for inclusion and linkage must be granted contractually by a user when repositing a document for publication.)

Legal accountability for both links and text will be the responsibility of their owners.

Private documents may be stored under the same system, but with slightly different rules: a private document may include or link to published documents, but not vice versa.

Storage can be distributed throughout the world and in space. Databases will not satisfy the information needs of the non-computer public, or the need to publish the ever-more-interconnected writings now being placed on electronic networks. It is our hope to bring the power of electronic access to a new and sweeping literary medium.

(Bibliography can be supplied on request.)

IV

THE OVERALL OBJECTIVE: UNIVERSAL HYPERTEXT PUBLISHING

These ideas may strain the imagination, but that's good. The unstretched imagination will atrophy.

The grand ambition of the Xanadu system is not "a database the size of the world," but rather a repository publishing network for anybody's documents and contents, which users may combine and link to freely.

This will permit a new form of electronic publishing, entirely within the tradition of paper publishing, but greatly streamlined: one need not ask permission to republish something, but simply place it in a document as an inclusion. Nothing will ever be misquoted or out of context, since the user may inquire as to the origins and native form of any quotation or other inclusion. Royalty will be automatically paid by a user whenever he or she draws out a byte from a published document. The permission structure of conventional publishing is bypassed. (Blanket permission for inclusion and linkage must be granted contractually by a user when reposting a document for publication.)

Legal accountability for both links and text will be the responsibility of their owners.

Private documents may be stored under the same system, but with slightly different rules: a private document may include or link to published documents, but not vice versa.

Storage can be distributed throughout the world and in space. Databases will not satisfy the information needs of the non-computer public, or the need to publish the ever-more-interconnected writings now being placed on electronic networks. It is our hope to bring the power of electronic access to a new and sweeping literary medium.